AUS920030792US1
Best et al.
Method and Apparatus for Increasing
Efficiency of Data Storage in a File System
1/4

EL9783174404S

100

104

108

102

106

110

FIG. 1

302 304 306 308

BB	SB	IL	DB

310 — FILE SYSTEM PROCESS

FIG. 3

CLIENT
200

| PROCESSOR | | HOST/PCI CACHE/BRIDGE | | MAIN MEMORY | | AUDIO ADAPTER |

202   208   204   216

BUS

206

| SCSI HOST BUS ADAPTER | LAN ADAPTER | EXPANSION BUS INTERFACE | GRAPHICS ADAPTER | AUDIO/ VIDEO ADAPTER |

212   210   214   218   219

DISK — 226

TAPE — 228

CD-ROM — 230

220 — KEYBOARD AND MOUSE ADAPTER

MODEM

MEMORY

222   224

FIG. 2

FIG. 4

```
400            402              406
FILE         INODE         EXTENDED
                             AREA

404    BLOCK

FILE       BLOCK

407         408         410      414      416   412
```

FIG. 5A

```
dinode {                                    500
        /*
        *       I. base area (128 bytes)
        *       -----------------------
        *
        *   define generic/POSIX attributes
        */
inostamp;  /* 4: stamp to show inode belongs to fileset */
fileset;       /* 4: fileset number */
number;    /* 4: inode number, aka file serial number */
gen;           /* 4: indode generation number */

ixpxd;             /* 8: inode extent descriptor */

size;              /* 8: size */
nblocks;   /* 8: number of blocks allocated */

nlink;             /* 4: number of links to the object */

uid;               /* 4: user id of owner */
gid;               /* 4: group id of owner */

mode;              /* 4: attribute, format and permission */

atime;     /* 8: time last data accessed */
ctime;     /* 8: time last status changed */
mtime;     /* 8: time last data modified */
otime;     /* 8: time created */

di_acl;            /* 16: acl descriptor */

di_ea;             /* 16: ea descriptor */

504 { next_index; /* 4: Next available dir_table index */
```
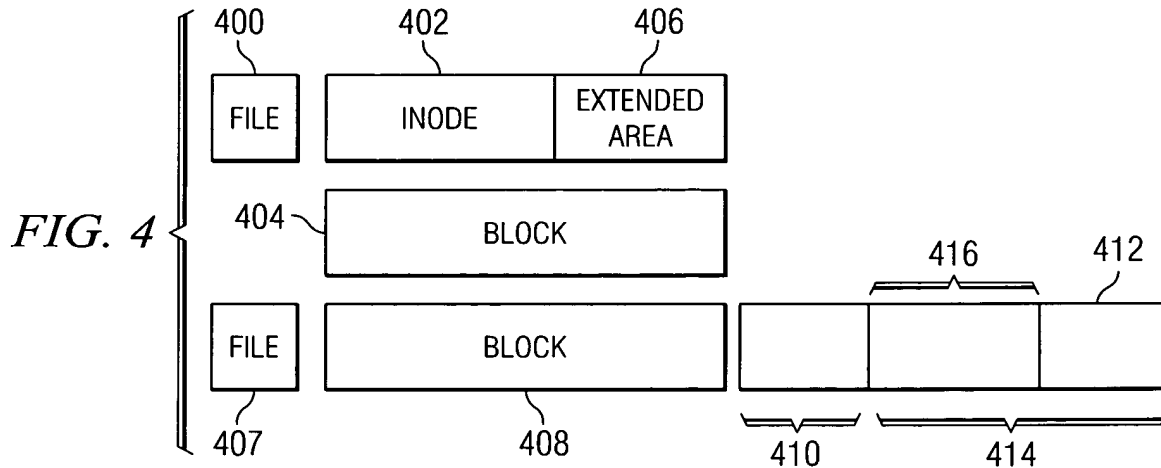
502

504

```
alctype;      /* 4:  Type of ACL */
      /*
      *      Extension Areas.
      *
      *      Historically, the inode was partitioned into 4 128-byte areas,
      *      the last 3 being defined as unions which could have mult
      *      uses. The first 96 bytes had been completely unused unt
      *      an index table was added to the directory. It is now mo
      *      useful to describe the last 3/4 of the inode as a single
      *      union.
      */
      union {
            struct {
                  /*
                  * This table contains the information needed to
                  * find a directory entry from a 32-bit index.
                  * If the index is small enough, the table is in
                  * otherwise, an x-tree root overlays this table
                  */
                  dir_table_slot_table[12] ;        /* 96: inline *

                  _dtroot;                     /* 288: dtree root */
            } _dir;                          /* (384) */

            struct {
                  union {
                        data [96] ;              /* 96: unused */
                        struct {
                              *_imap;        /* 4: unused */
                              _gengen ;      /* 4: generator */
                        } _imap;
                  } _ul;                              /* 96: *

                  union {
                        _xtroot[288] ;
                        struct {
                              unused[16] ;    /* 16: */
                              _dxd;           /* 16: */
                              union {
                                          dev;          /* 4: */
                                          _fastsymlink [128] ;
                              } _u;
                              _inlineea[128] ;
                              _inlinedata[128] ;
                        } _special;                508
                  } _u2;
            } _file;
      } u;
};
```
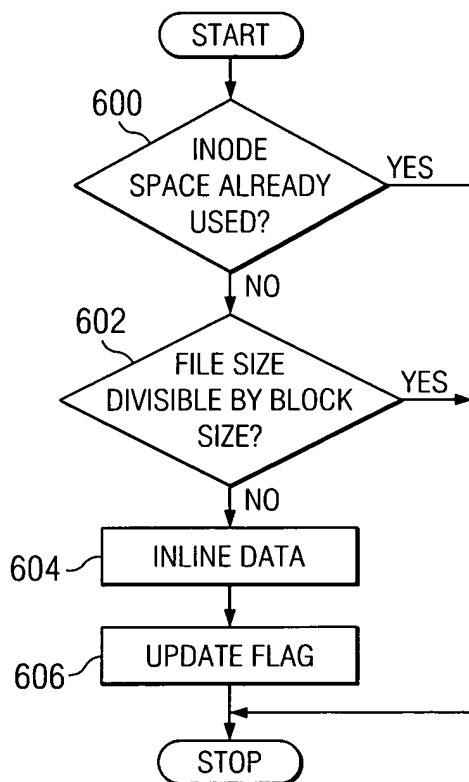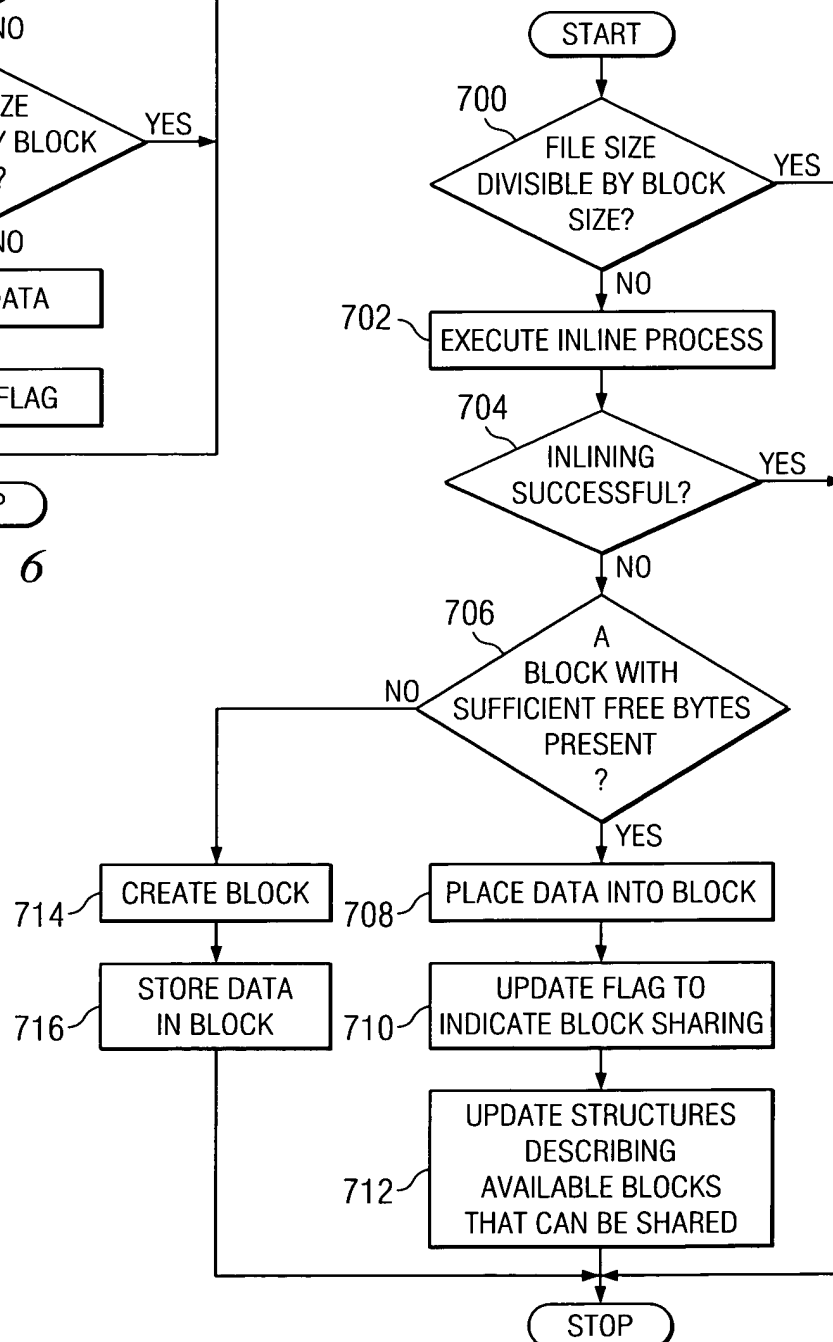
506

508

FIG. 5B

AUS920030792US1
Best et al.
Method and Apparatus for Increasing
Efficiency of Data Storage in a File System
4/4

START

600 INODE SPACE ALREADY USED? — YES

NO

602 FILE SIZE DIVISIBLE BY BLOCK SIZE? — YES

NO

604 INLINE DATA

606 UPDATE FLAG

STOP

FIG. 6

START

700 FILE SIZE DIVISIBLE BY BLOCK SIZE? — YES

NO

702 EXECUTE INLINE PROCESS

704 INLINING SUCCESSFUL? — YES

NO

706 A BLOCK WITH SUFFICIENT FREE BYTES PRESENT? — NO

YES

714 CREATE BLOCK

708 PLACE DATA INTO BLOCK

716 STORE DATA IN BLOCK

710 UPDATE FLAG TO INDICATE BLOCK SHARING

712 UPDATE STRUCTURES DESCRIBING AVAILABLE BLOCKS THAT CAN BE SHARED

STOP

FIG. 7